Section 16

System Design

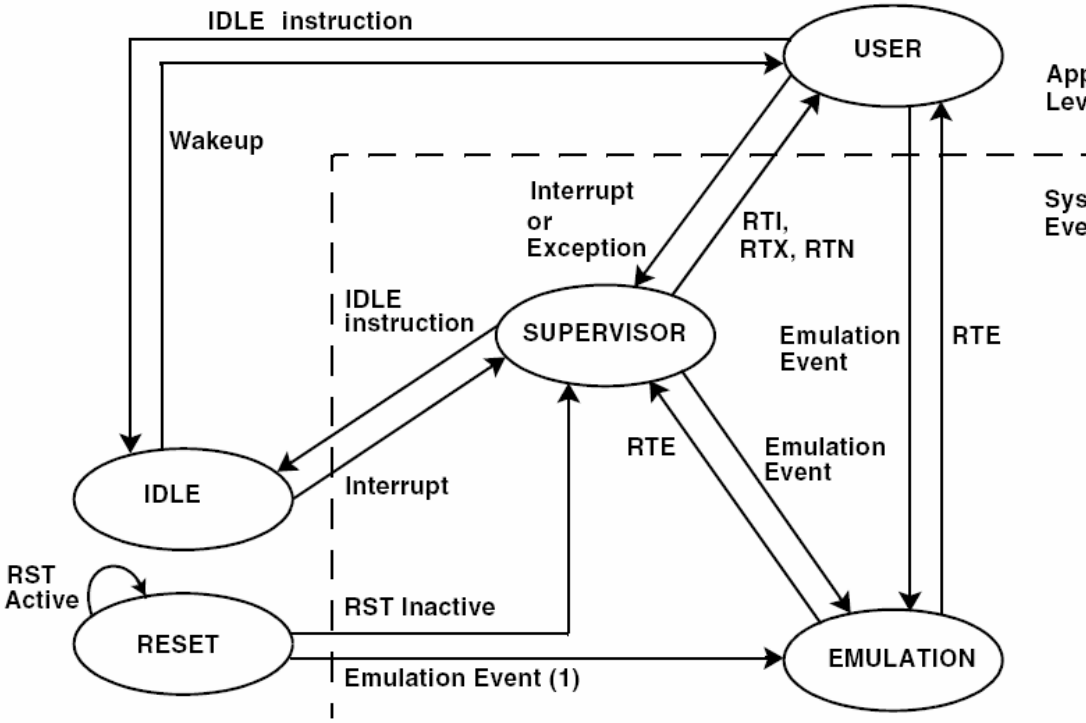
Operating Modes

Operating Modes

- **User mode**
  Causes exceptions when protected resources are accessed.
  May be used for algorithm/application code
- **Supervisor mode**
  has unprotected access to all resources.
  May be used for O/S kernel, device drivers, debuggers, ISRs
- **Emulator (or Debug) mode**
  has supervisor abilities and is accessible via JTAG

Operating Modes provide a feature to implement RTOS architectures and Multitasking schemes. Smaller applications may simply run in Supervisor mode all the time.

Operating Modes



(1) Normal exit from Reset is to Supervisor mode. However, emulation hardware may have initiated a reset. If so, exit from Reset is to Emulation.

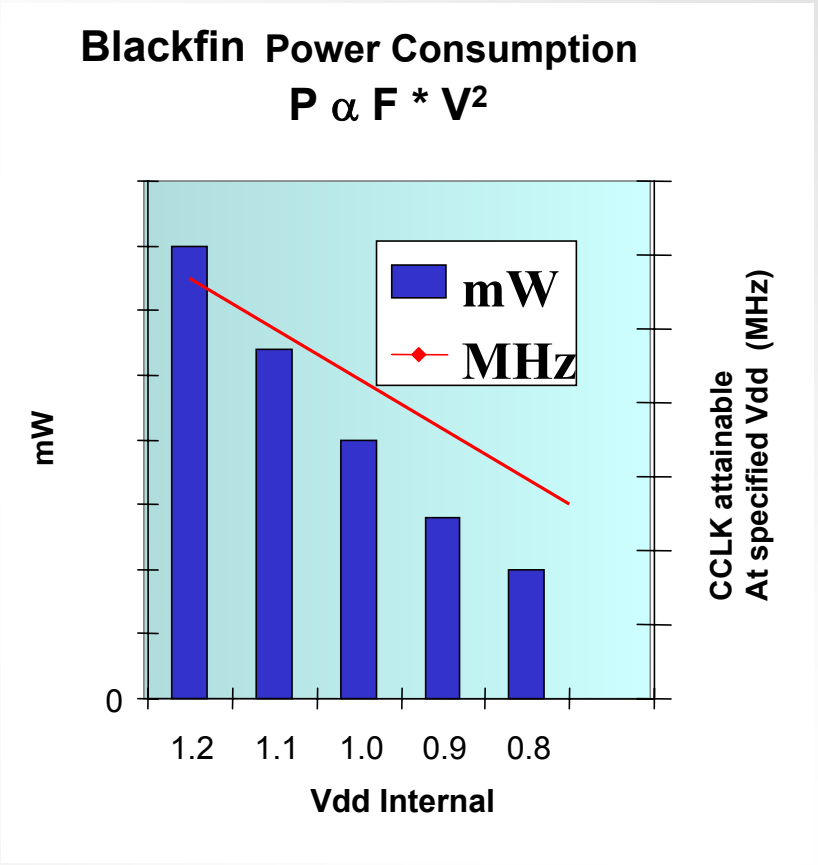
Dynamic Power Management

Power Management Options

- Low Active Power
  - Flexible power management with automatic power-down for unused peripheral sections
  - Dynamic Power Management allows dynamic modification of both frequency and voltage
- Low Standby Power
  - 4 Power modes
  - Real Time Clock with alarm and wakeup features

Dynamic Power Management

- Clocks to unused peripherals are automatically disabled.
- Integrated Switching Regulator Controller
  - SW-based Voltage-Scaling Capability



Blackfin DSPs
Control Voltage & Frequency

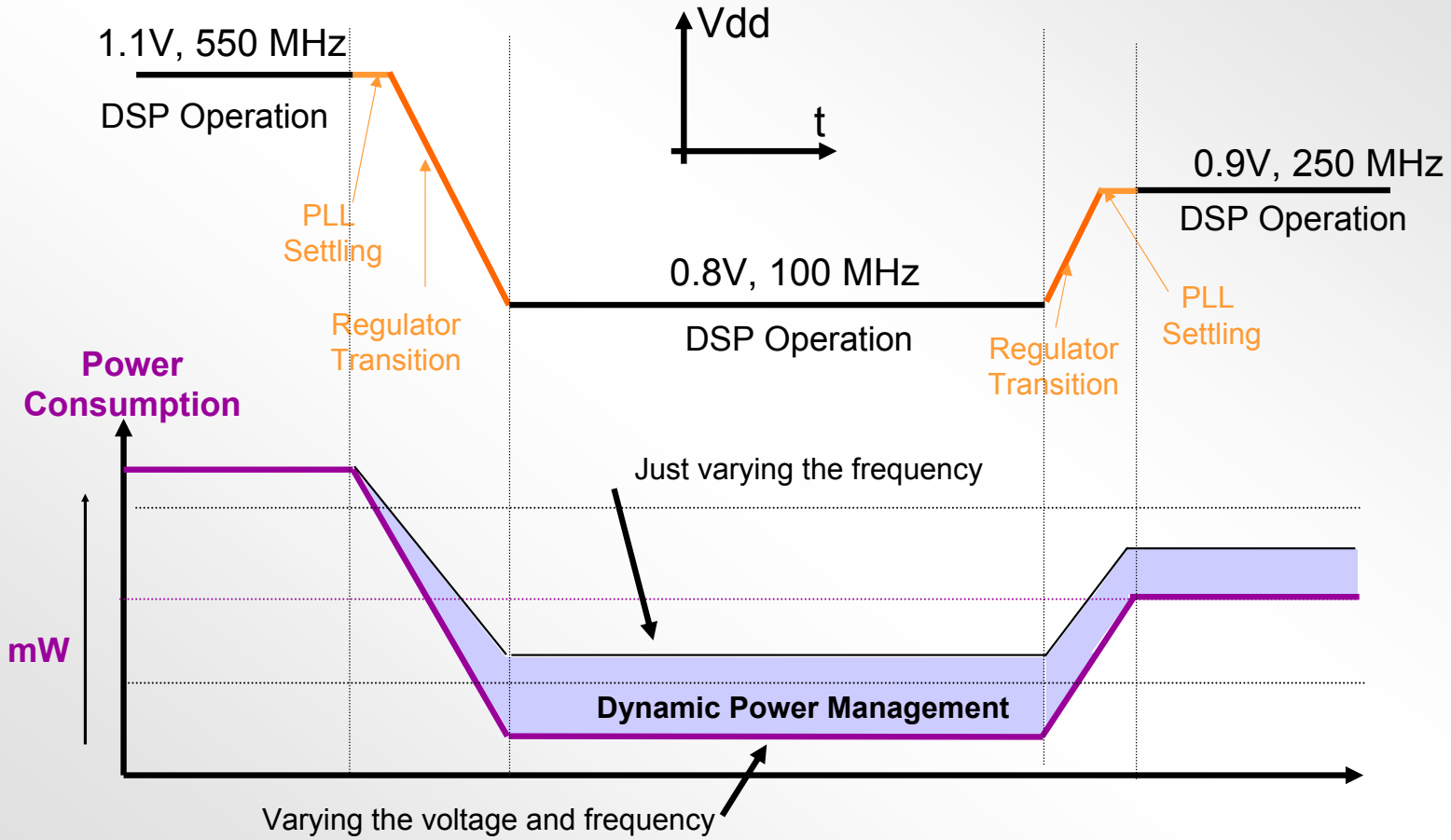
- **Onboard, software-controlled switching regulator controller**
- **Highly flexible 1x-64x PLL allows easy frequency scaling**
- **Multiple Power-Down Modes**
- **Functional & Peripheral Blocks are Clocked Only When Used**

Dynamic Power Management Using RTOS or Firmware

Function	Example MHz	Example Vdd (V)
$F_0(x)$	550	1.1
$F_1(y)$	250	0.95
$F_n(z)$	100	0.8

Profiling Tools Audit MIPS Requirements By Function

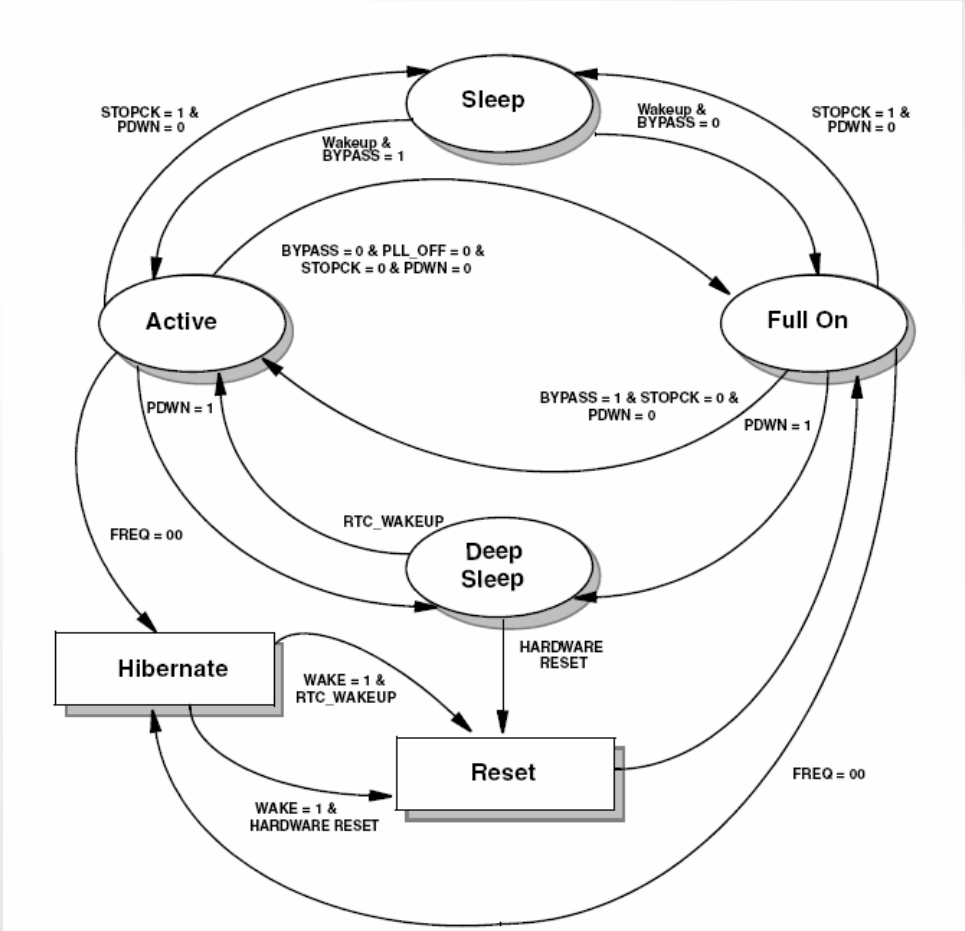
Blackfin DSPs
Optimize Power Consumption



Power Management States

Mode	Relative Power Savings	Notes
Full On	**Min**	**Max performance**
Active	**Low**	**Full core operation at CLKIN.** **System DMA to L1 supported.** **PLL is bypassed and can be disabled.**
Sleep	**High**	**Core idle. CCLK disabled. SCLK enabled.**
Deep Sleep	**Very High**	**Core idle. Only Real-Time Clock enabled.** **Exit only via HW reset or RTC interrupt.**
Hibernate	**Max**	**VDDINT is disabled. Only VDDEXT applied.** **Power up via HW reset or RTC interrupt.**

Power Mode Transitions



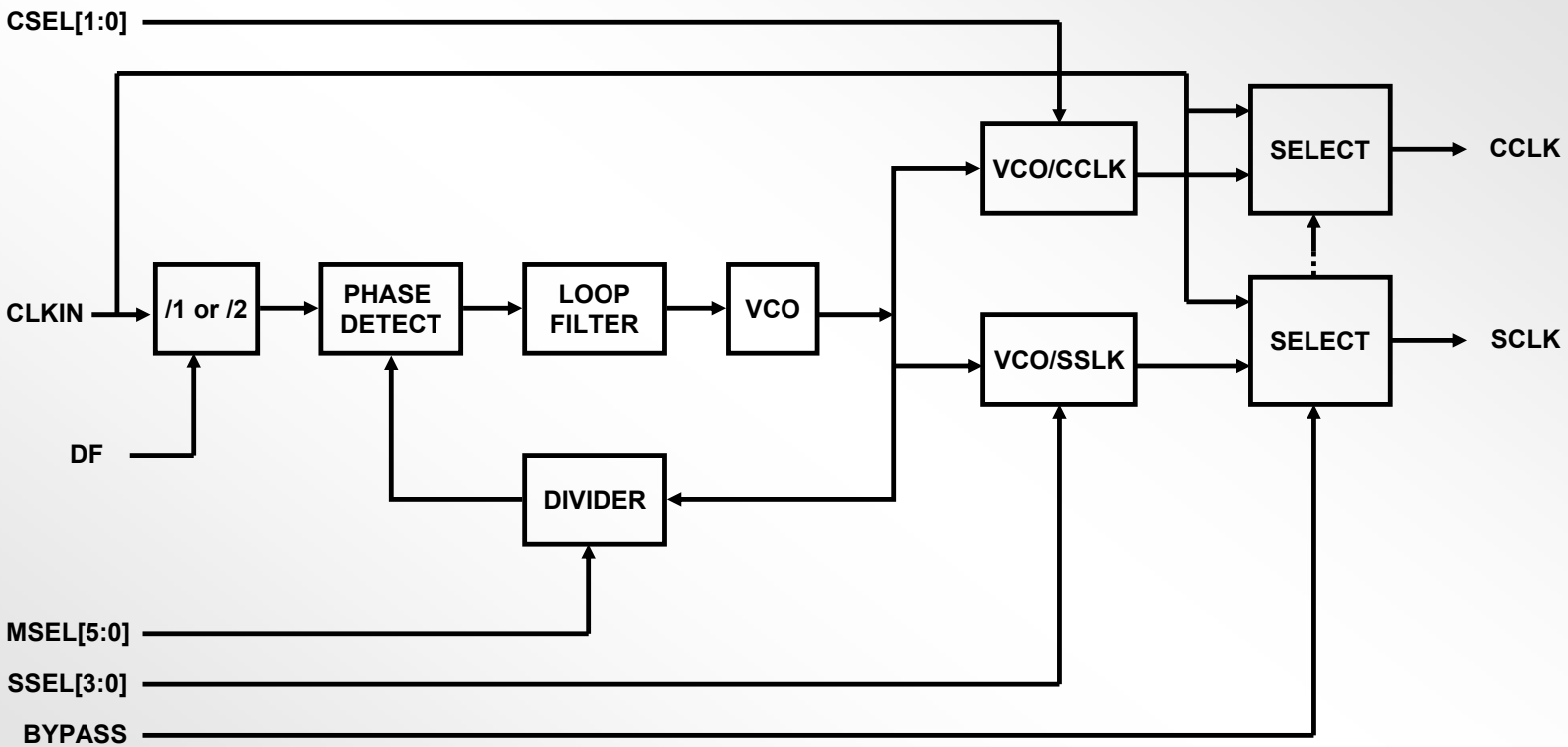
Clock States in Different Power Modes

Mode	PLL	PLL Bypassed?	Core Clock (CCLK)	System Clock (SCLK)
Full On	Enabled	No	Enabled	Enabled
Active	Enabled or Disabled	Yes	Enabled	Enabled
Sleep	Enabled	No	Disabled	Enabled
Deep Sleep *Hibernate*	Disabled	---	Disabled	Disabled

BF533 Clocking

- The BF533 has 2 internal clock domains: CCLK (core clock) and SCLK (system clock)
  - CCLK is divided down from the PLL VCO frequency (via CSEL bits in PLL_DIV), or equals the CLKIN pin frequency if the PLL is bypassed
  - SCLK is divided down from the PLL VCO frequency (via SSEL bits in PLL_DIV), or equals the CLKIN pin frequency if the PLL is bypassed
    - SCLK must not exceed 133 MHz
- CLKIN can be driven from external oscillator or crystal
- Programmable PLL supports 1x to 64x frequency multiplication, enabling high-speed operation with low-frequency clock inputs
  - Program via bits in PLL_CTL register

Phase-Locked Loop Architecture



CCLK Ratio Control

Signal Name	Divider Ratio	Example Frequency Ratios (MHz)	
CSEL[1:0]	VCO/CCLK	VCO	CCLK
00	1	300	300
01	2	600	300
10	4	600	150
11	8	400	50

- **Upon reset, CSEL[1:0] = 00**

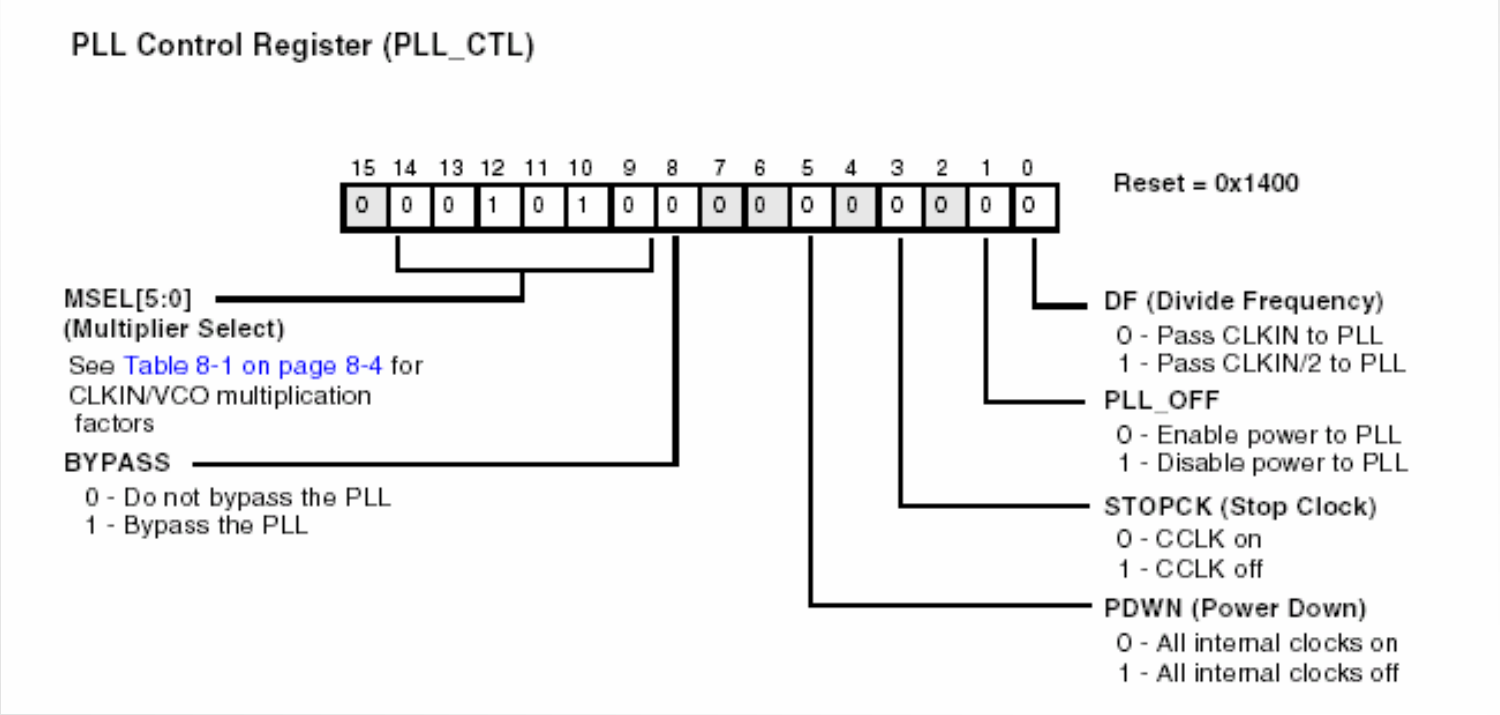
SCLK Ratio Control

Signal Name	Divider Ratio	Example Frequency Ratios (MHz)	
SSEL[3:0]	VCO/SCLK	VCO	SCLK
0000	Reserved	N/A	N/A
0001	1:1	100	100
0010	2:1	200	100
0011	3:1	400	133
0100	4:1	500	125
0101	5:1	600	120
0110	6:1	600	100
N=7-15	N:1	600	600/N

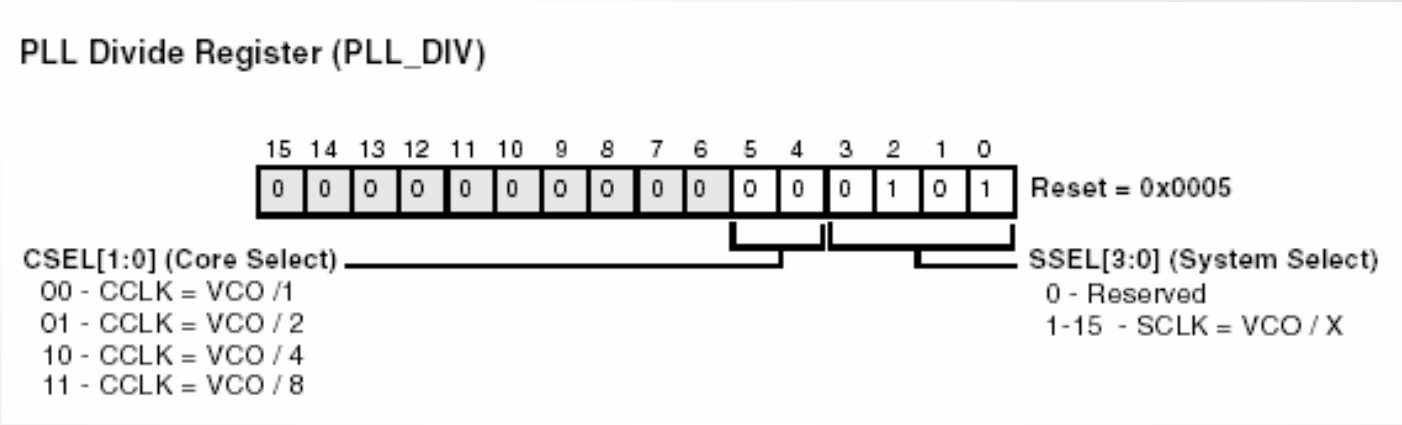
- **Upon reset, SSEL[3:0] = 0101**

PLL Control Register (PLL_CTL)

- **Controls operation of the PLL, specifying loop parameters and global control bits**



PLL Divide Register (PLL_DIV)



- **Programmer must ensure SCLK frequency always less than or equal to CCLK frequency**
  - **Otherwise, SCLK will be automatically adjusted to fall into compliance, but not necessarily optimized for top allowable speed**
- **Programmer must ensure SCLK frequency does not exceed 133 MHz**

Programming PLL Transitions

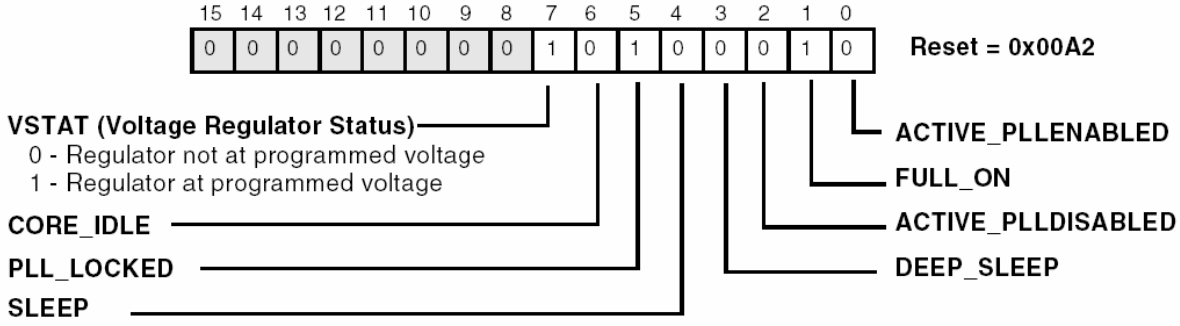
- Simply modifying PLL_CTL bits will not change the PLL operating mode until a specific code sequence executes:
  - CLI R0; /* disable interrupts */
  - IDLE; /* drain pipeline and send core into idle state */
  - STI R0; /* re-enable interrupts after wakeup */
- This sequence is necessary when changes have been made to MSEL, DF, or operating state bits (PDWN, BYPASS, STOPCK)
  - However, changes to CSEL or SSEL divide ratios take effect immediately, without needing the above sequence
- If the CLKIN-to-VCO multiplier has been changed, or the PLL has been re-enabled, the PLL will now need to relock

PLL Status Register (PLL_STAT)

- **PLL_STAT indicates the operating mode of the PLL and the ADSP-BF533 processor**

PLL Status Register (PLL_STAT)

Read only. Unless otherwise noted, 1 - Processor operating in this mode. For more information, see "Operating Modes" on page 8-14.



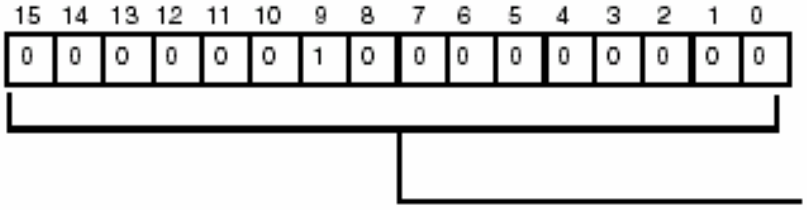
Relocking the PLL

- PLL lock count starts once the IDLE instruction has executed
- PLL lock counter is cleared and then increments each SCLK cycle
- When it reaches the value programmed into the PLL_LOCKCNT MMR, the PLL_LOCKED bit is set in PLL_STATUS
- Then the PLL Wakeup interrupt is asserted, and an interrupt will occur if this interrupt is enabled in SIC_IMASK

PLL Lock Count Register (PLL_LOCKCNT)

- PLL_LOCKCNT defines the number of SCLK cycles before the PLL_LOCKED bit (in PLL_STAT) gets set after a PLL transition

PLL Lock Count Register (PLL_LOCKCNT)



Reset = 0x0200

LOCKCNT
Number of SCLK cycles before PLL Lock Count timer expires.

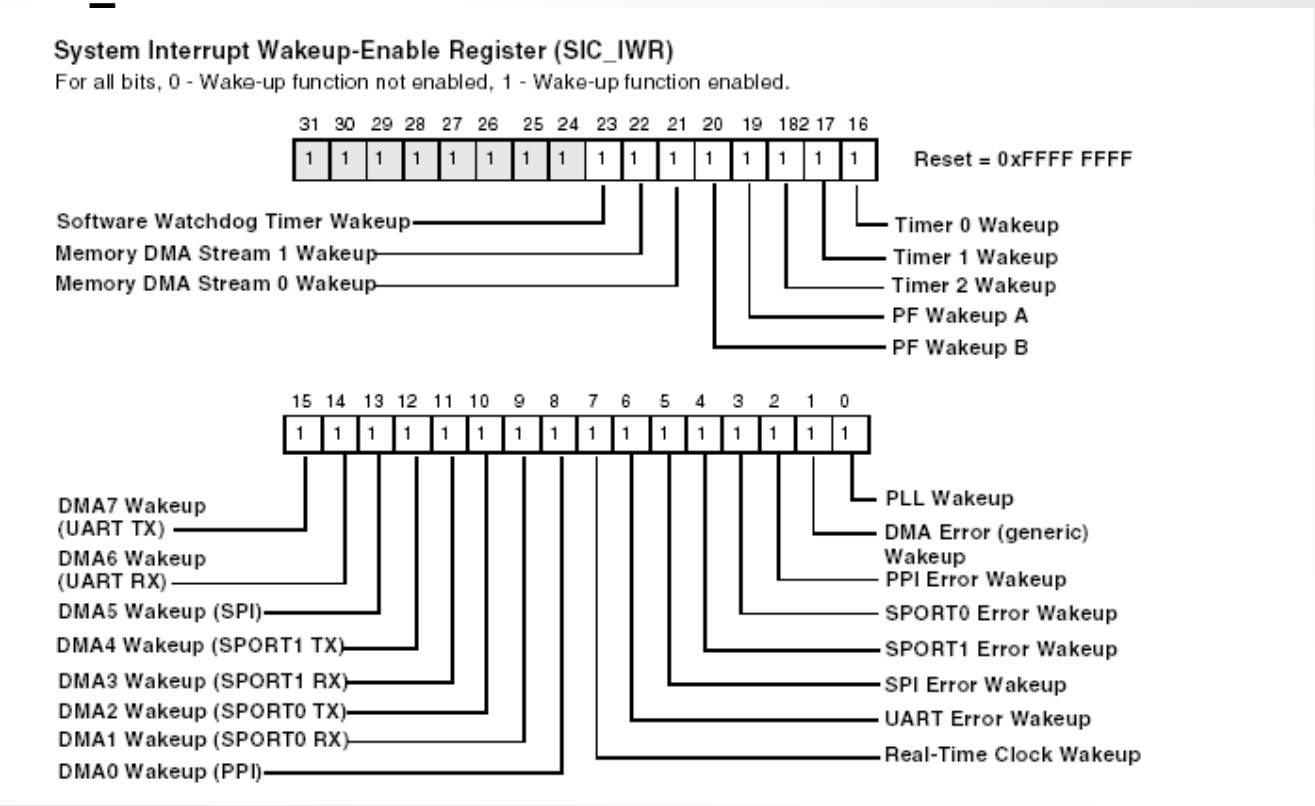
- Can generate system wakeup (in SIC_IWR) when lock count reached

IDLE state

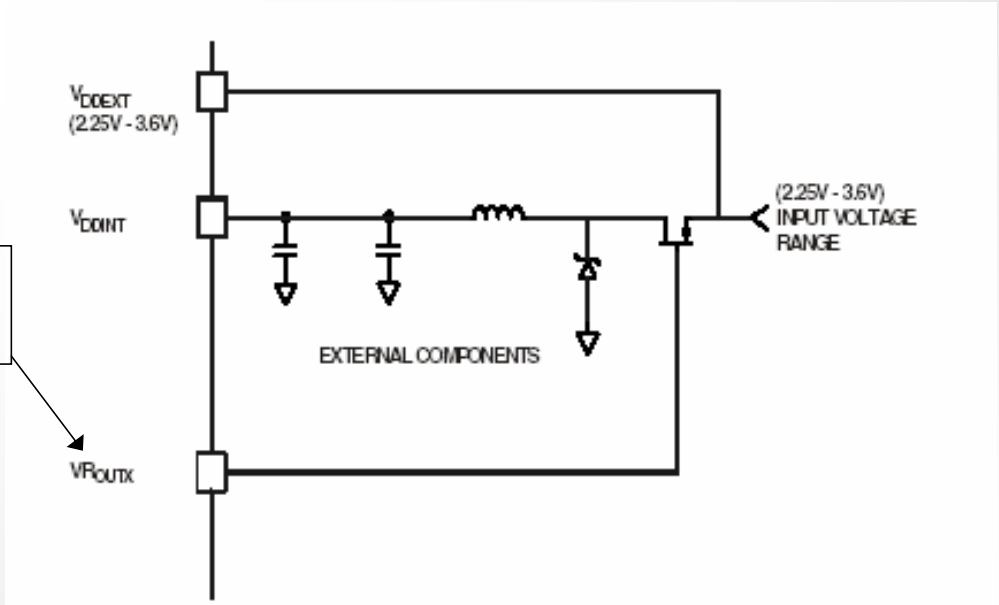
- After executing ‘IDLE’, an ‘SSYNC’ instruction automatically occurs.
  - The DSP core stops executing instructions, retains the contents of pipeline and waits for an interrupt or wakeup.
  - PLL, CCLK and SCLK continue running
- 2 ways to leave an IDLE state
  - DSP services an interrupt. DSP will return to the instruction following the IDLE after executing the RTI instruction.
  - A peripheral wakes the DSP up (based on SIC_IWR settings), but no interrupt occurs. DSP returns to instruction that follows IDLE

System Interrupt Wakeup-Enable Register (SIC_IWR)

- **Choose the peripherals that can wake the core from an idle state in SIC_IWR**



On-chip Voltage Regulation

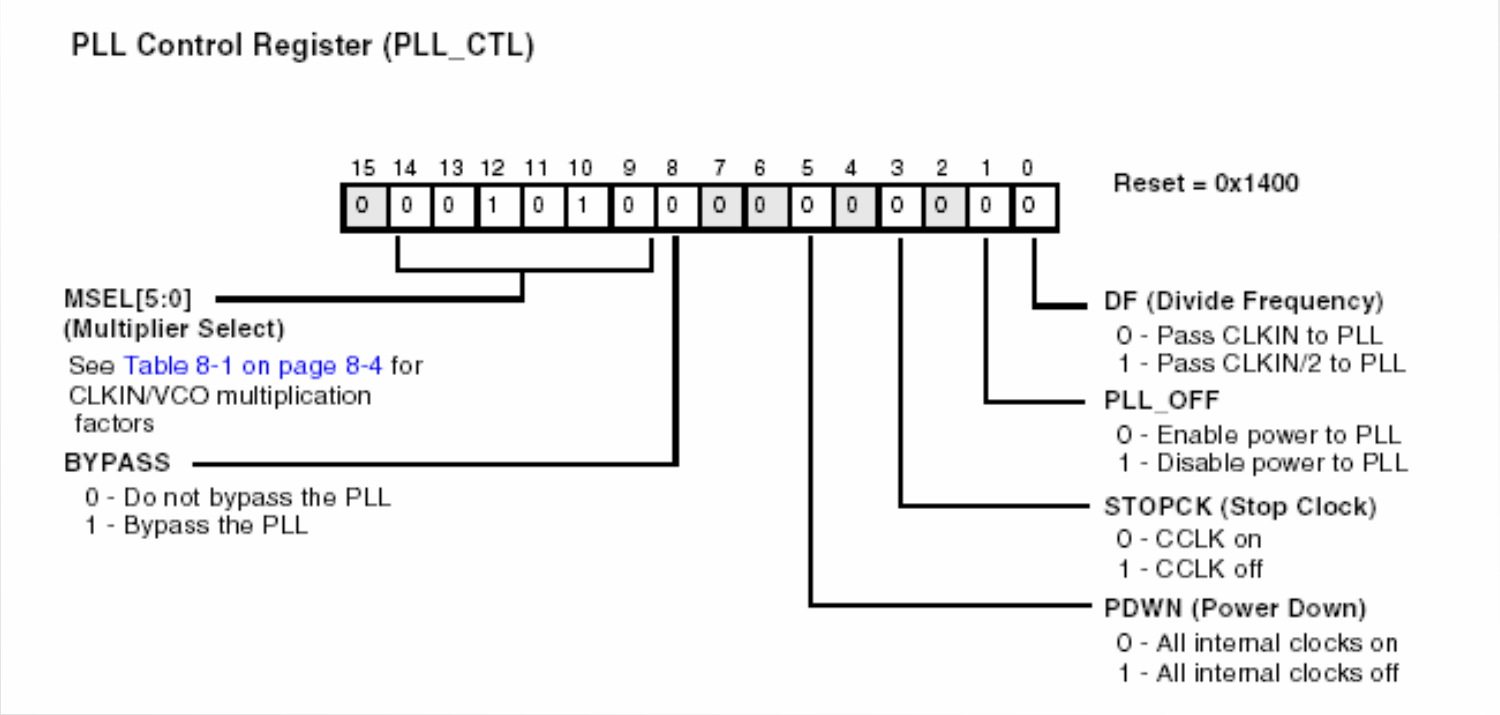


When using on-chip regulator, tie both Vrout pins together

- BF533/BF561 has internal ***switching regulator controller***
  - ***NOT*** a linear regulator
  - External FET, Diode, L, C must be supplied
  - 2.25V – 3.6V input range regulated down to VDDint (0.8V-1.2V)

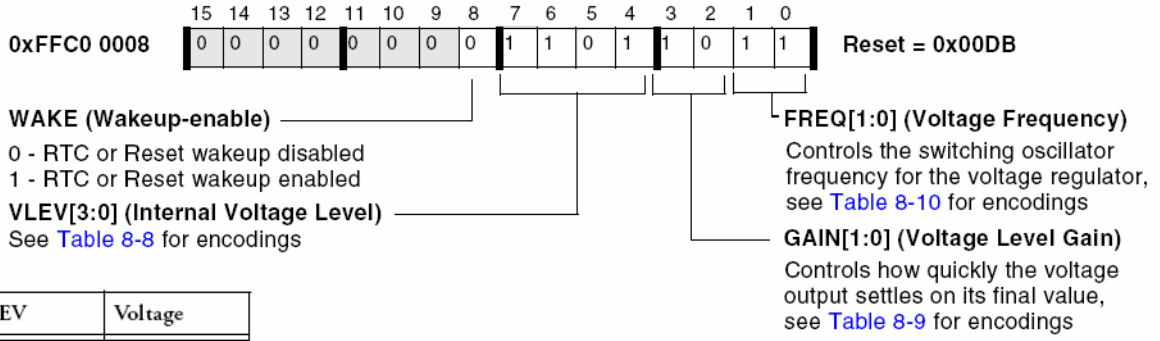
PLL Control Register (PLL_CTL)

- **Controls operation of the PLL, specifying loop parameters and global control bits**



ADSP-BF533 Voltage Regulator Control Register (VR_CTL)

Voltage Regulator Control Register (VR_CTL)



VLEV	Voltage
0000–0101	Reserved
0110	.85 volts
0111	.90 volts
1000	.95 volts
1001	1.00 volts
1010	1.05 volts
1011	1.10 volts
1100	1.15 volts
1101	1.20 volts
1110	1.25 volts
1111	1.30 volts

FREQ	Value
00	Powerdown
01	333KHz
10	667KHz
11	1MHz

Gain	Value
00	5
01	10
10	20
11	50

To bypass the on-chip regulator

- Leave the 2 VROUT pins floating
- Set FREQ[1:0] in VR_CTL to ’00’
- Connect external 0.8V-1.2V supply to VDDint pins

Resetting the ADSP-BF533

Reset Types

Reset	Source	Result
Hardware reset	The $\overline{\text{RESET}}$ pin causes a hardware reset.	Resets both the core and the peripherals, including the Dynamic Power Management Controller (DPMC). Resets the No Boot on Software Reset bit in SYSCR. For more information, see "System Reset Configuration Register (SYSCR)" on page 3-14.
System Software reset	Writing b#111 to bits [2:0] in the system MMR SWRST at address 0xFFC0 0100 causes a System Software reset.	Resets only the peripherals, excluding the RTC (Real-Time Clock) block and most of the DPMC. The DPMC resets only the No Boot on Software Reset bit in SYSCR. Does not reset the core.
Watchdog Timer reset	Programming the watchdog timer appropriately causes a Watchdog Timer reset.	Resets both the core and the peripherals, excluding the RTC block and most of the DPMC. The Software Reset register (SWRST) can be read to determine whether the reset source was the watchdog timer.
Core Double-Fault reset	If the core enters a double-fault state, a reset can be caused by unmasking the Core Double-Fault Reset Mask bit in the System Interrupt Controller (SIC) Interrupt Mask register (SIC_IMASK).	Resets both the core and the peripherals, excluding the RTC block and most of the DPMC. SWRST can be read to determine whether the reset source was Core Double-Fault.
Core-Only Software reset	Executing a RAISE1 instruction or by setting the Software Reset (SYSRST) bit in the core Debug Control register (DBGCTL) via emulation software through the JTAG port. DBGCTL is not visible to the memory map.	Resets only the core. The peripherals do not recognize this reset.

System transitions into boot mode sequence upon completion of internal reset

Occurs when an exception is generated while another exception is being handled

•Both of these SW resets are used primarily for debugging purposes.

•The system may be in an unreliable state after a SW reset period ends. Therefore, execution must be from L1 memory after a Core or Peripheral SW Reset.

Hardware Reset

- Asynchronous
- /RESET pin asserted low until after supplies have stabilized
- After pin deasserted, reset timer allows all peripherals to complete a reset
- Interrupt request generated
- System then moves into boot mode (based on BMODE pins)

Staying in Supervisor Mode after Reset

- If executing from external memory after hardware reset, the ADSP-BF533 will be in the Reset ISR
- This implies the part is in Supervisor mode, but…
  - Lower priority events can not be serviced until you return from Reset ISR
  - Customer can force the lowest priority interrupt in order to remain in Supervisor mode
- If booting after hardware reset, the above process is automatically performed by the Boot ROM

Supervisor Mode Reset Example

```
/************************Code to Stay in Supervisor Mode*************************************/
SP.H = 0xF003;              //Set up supervisor stack
SP.L = 0xFFDC;
P0.L =LO(EVT15);            //Point to EVT15 in Event Vector Table
P0.H = HI(EVT15);
P1.L = START;               //Point to start of Boot Rom code (or any other code)
P1.H = START;
[P0] = P1;                  //Place the address of start code in EVT15 of Event Vector Table
P0.L = LO(IMASK);
R0 = W[P0];
R1.L = LO(EVT15);
R0 = R0 | R1;
W[P0] = R0;                 //Set(enable) EVT15 bit in IMASK Register
RAISE 15;                   //Invoke ET15 interrupt (still in higher-priority ISR, so nothing happens yet)
P0.L = WAIT_HERE;
P0.H = WAIT_HERE;
RETI = P0;
RTI;                        //Return from Reset Interrupt to allow processing of lower priority interrupts
WAIT_HERE:                  //Wait here till EVT15 interrupt is processed
JUMP WAIT_HERE;
/*********************************************************************************************/
START:
[--SP] = RETI;              // Re-enable interrupts
```

Used to re-enter Supervisor mode

Default Settings after Reset

Item	Description of Reset State
Core	
Operating mode	Supervisor mode in reset event, clocks stopped
Rounding mode	Unbiased rounding
Cycle counters	Disabled, zero
DAG registers (I,L,B,M)	Random values (must be cleared at initialization)
Data and Address registers	Random values (must be cleared at initialization)
IPEND, IMASK, ILAT	Cleared, interrupts globally disabled with IPEND bit 4
CPLBs	Disabled
L1 Instruction Memory	SRAM (cache disabled)
L1 Data Memory	SRAM (cache disabled)
Cache validity bits	Random (must be set to invalid before cache initialization)
System	
Booting methods	Determined by the values of BMODE pins at reset
MSEL clock frequency	Reset value = 10
PLL Bypass mode	Disabled
VCO/Core Clock Ratio	Reset value = 1
VCO/System Clock Ratio	Reset value = 5
Peripheral clocks	Disabled

PLL Settings after Reset (Example)

Assume CLKIN = 25MHz Oscillator

MSEL[5:0] = 0x0A, DF = 0

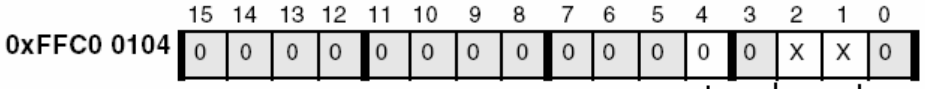
VCO = 25MHz x 10 = 250MHz

CCLK = VCO/1 = 250 MHz

SCLK = VCO/5 = 50 MHz

BF533 System Configuration Reset Register (SYSCR)

System Reset Configuration Register (SYSCR)
X - state is initialized from mode pins during hardware reset

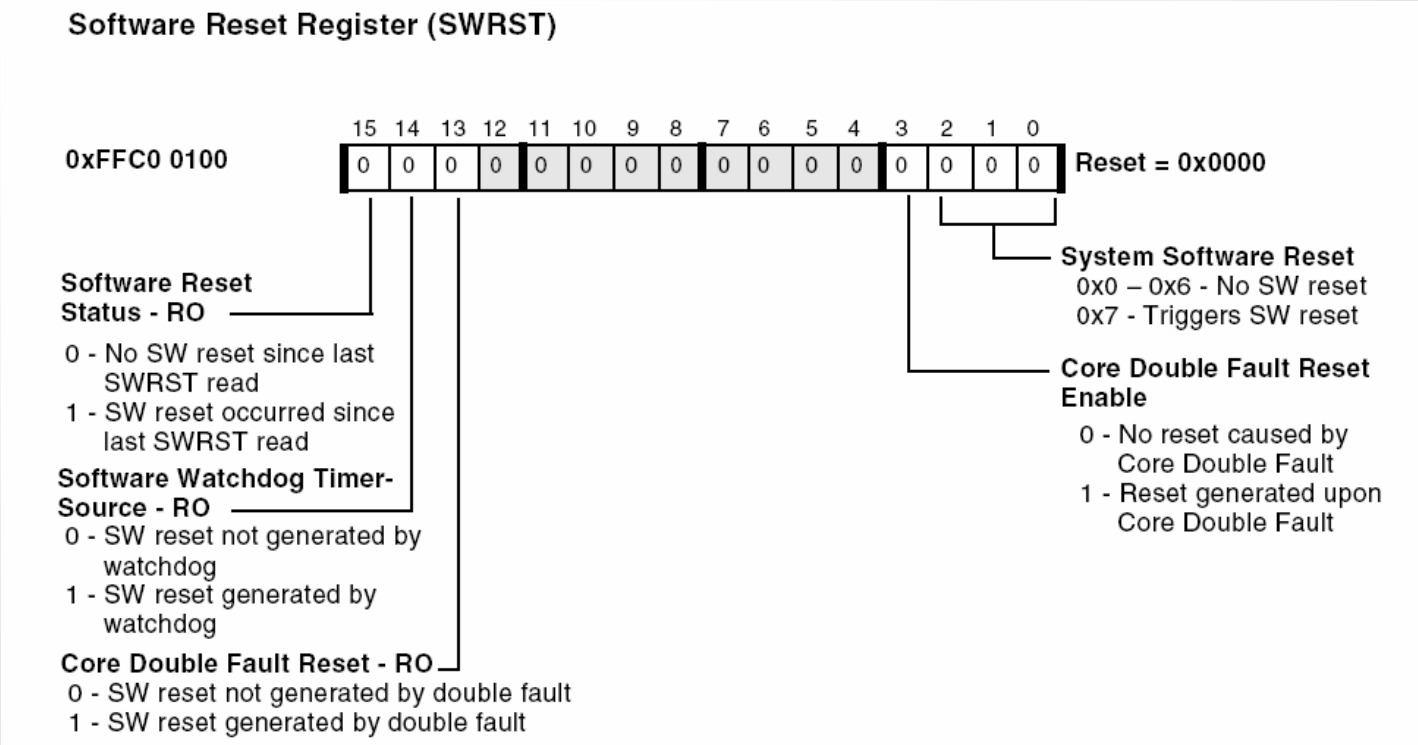


Reset = dependent on pin values

No Boot on Software Reset
0 - Use BMODE to determine boot source
1 - Start executing from the beginning of on-chip L1 memory or the beginning of ASYNC Bank 0 when BMODE[1:0] = b#00

BMODE[1:0] (Boot Mode)- RO
00 - Bypass boot ROM, execute from 16-bit external memory
01 - Use boot ROM to load from 8-bit or 16-bit flash
10 - SPI slave mode boot via a master (host)
11 - Use boot ROM to configure and load boot code from SPI serial EEPROM (8-, 16-, or 24-bit address range)

Software Reset Register (SWRST)



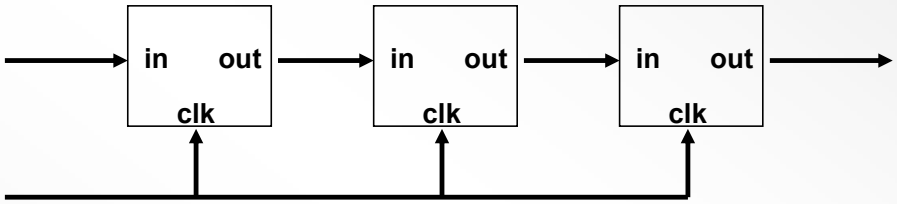
JTAG Port

In-Circuit Emulator Support

- 5-pin JTAG TAP (Test Access Port) allows access to JTAG features
- TAP Controller handles test register event sequencing
- Boundary scan facilitates board-level connectivity testing
  - Up to 32 boundary-scan instructions accommodated

ADSP-BF533 JTAG

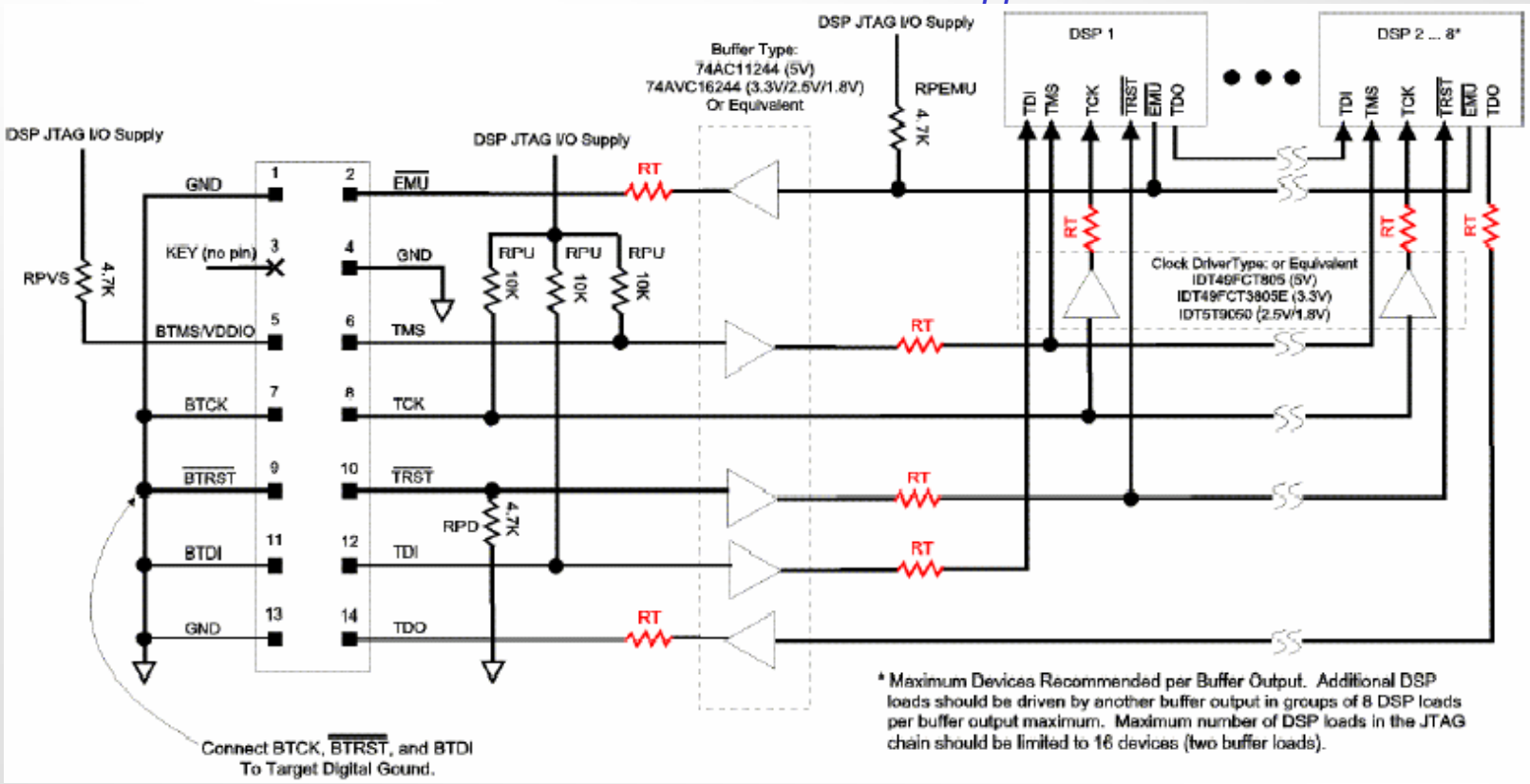
- IEEE standard 1149.1-1990
- Boundary Scan (bsdl files available)



- Read Part Identification Code
  - Analog Devices JEDEC ID is 0x0E5
  - Instruction Register is 5 bits wide
- In Bypass mode behaves like a 1-bit shift register
- Emulation (ADI enhancement)

ADSP-BF533 JTAG Emulation

- **Provides communication channel between JTAG emulator hardware and ADI JTAG DSP.** *See Application Note EE-68*



Board Design and Layout

What do you need for ADSP-BF533/BF561 Hardware Design?

- ADSP-BF533 DSP Hardware Reference
  - Includes Hardware Examples
- The Datasheet (check the web for latest revision!)
  - Insure that all timing requirements are met!!!!
- The Anomaly Sheet (check the web for latest revision!)
- Use EZ-Kits as Design Example
- Look for Application Notes
- Recommended Reading

  High Speed Digital Design - A Handbook of Black Magic
  Johnson & Graham
  Prentice Hall,Inc.
  ISBN 0-13-395724-1

Navigating the Datasheet

- Power Supply and Input Voltage and Temperature Ranges
  - ABSOLUTE MAXIMUM RATINGS
    (When violated, device might be damaged)
  - RECOMMENDED OPERATING CONDITIONS
    (When violated, device might not work properly)
- ELECTRICAL CHARACTERISTICS
  - Tested or guaranteed DC characteristics
- TIMING PARAMETERS
  - How do Interface Output Pins behave
  - What do Interface Input Pins require
- ADDITIONAL DIAGRAMS
  - Typical behavior of power consumption and I/O stages
  - No guaranteed limits !

Powering Up the DSP

- The ADSP-BF533 has multiple power domains.
- Apply the core supply (VDDINT) first and limit supply current as well as the duration of unpowered I/O to prevent latch-up effects and damage of isolating diodes.
- Never apply external clock to CLKIN pin while DSP is not powered.

Power Domain	V_{DD} Range
All internal logic	0.8V(min) -1.2V ± 5%
All I/O	2.5-3.3V ± 10%

ADSP-BF533 Internal Current Consumption

- Core Supply IDDINT
  - Peak
  - Executing from internal memory
    - TBD
  - Typical
  - Executing from internal memory
    - 50% are MAC instruction
  - Power-down All
  -
- Peripheral Supply IDDINT
  - Typical
  - Power-down

ALWAYS CHECK DATA SHEET FOR LATEST INFO.

Calculate Power Dissipation

ALWAYS REFER TO DATA SHEET FOR CURRENT SPECIFICATIONS

Total Power Dissipation $\overline{P_{Total}} = \overline{P_{Intern}} + \overline{P_{Extern}}$

Internal Power Dissipation $\overline{P_{Intern}} = V_{DDIN} \cdot \overline{I_{DDIN}}$

V_{DDINT} = 1.2V

Typical I_{DDIN} is specified in the Datasheet and depends on operating mode

External Power Dissipation $\overline{P_{Extern}} = V_{DDEXT}^{2} \cdot \sum_{Pins} f \cdot C$

V_{DDEXT} = 3.3V

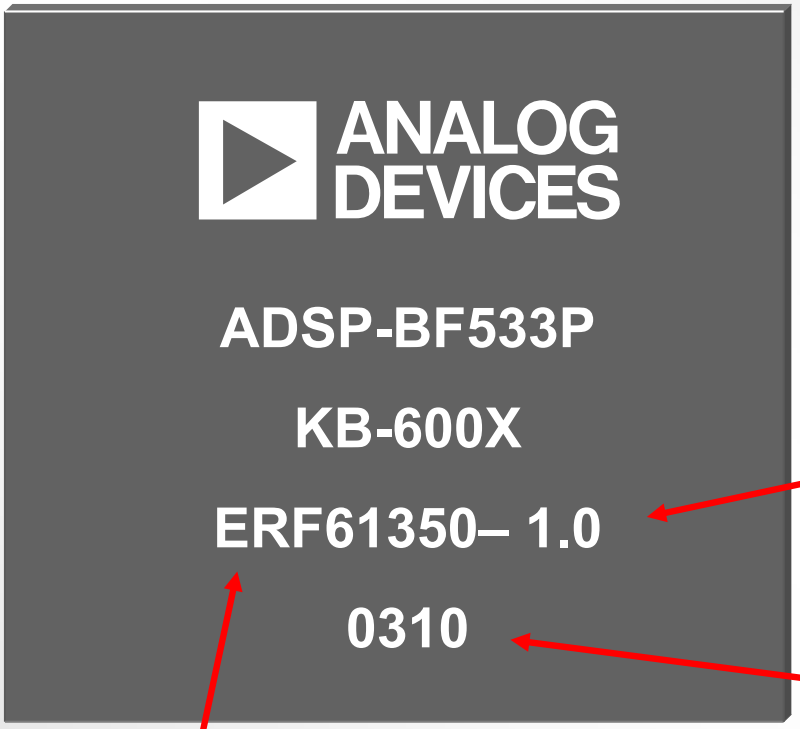
Capacitive Bus Load

(BF533: C_{IN} max = x pF)

High-Speed Design Issues

- Awareness of High Speed Digital Design
  - Signal Integrity on Wires and Connectors
  - Noise generated by Crosstalk
  - EMI / EMC Compliance
- Bypassing Capacitors (Ceramic)
  - 100nF decoupling capacitors between VddInt and Gnd
  - 100nF decoupling capacitors between VddExt and Gnd
  - One 100nF capacitor at the power supply connector of the board
- Grounding and Shielding
  - Dedicated VDD and GND supply plane recommended
  - No Ground Loops
  - Signal Return Path as short as possible
  - No Signal Routing over Split Planes

Information printed on the package



Chip Revision 1.0

Date Code:
10th week of 2003

Lot ID

"E" Wafer Fab Code = TSMC Taiwan

"R" Assembly Location Code = STATS Singapore

Debug Registers

Advanced Support for Embedded Debug

- Hardware Breakpoints
  - 6 Instruction and 2 Data Watchpoints
- Performance Monitor Unit
  - Counters for cycles and occurrences of specific activities
- Execution Trace Buffer
  - Stores last 16 non-incremental PC values

Reference Material

System Design

Hardware Breakpoints

- 8 Watchpoints for Instruction/Data Address Comparison
  - 6 Instruction Watchpoints, 2 Data Watchpoints
  - Can be used as 4 Watchpoint ranges instead (3 instruction address ranges + 1 data address range)
- Allow conditional program halting, based on user-specified conditions
  - Memory accesses within a specified range
  - Data loaded into register
  - Stack activity
- Counter allows tracking of watchpoint events
- Ability to combine events
  - Break ONLY when ALL or when ANY enabled events occur

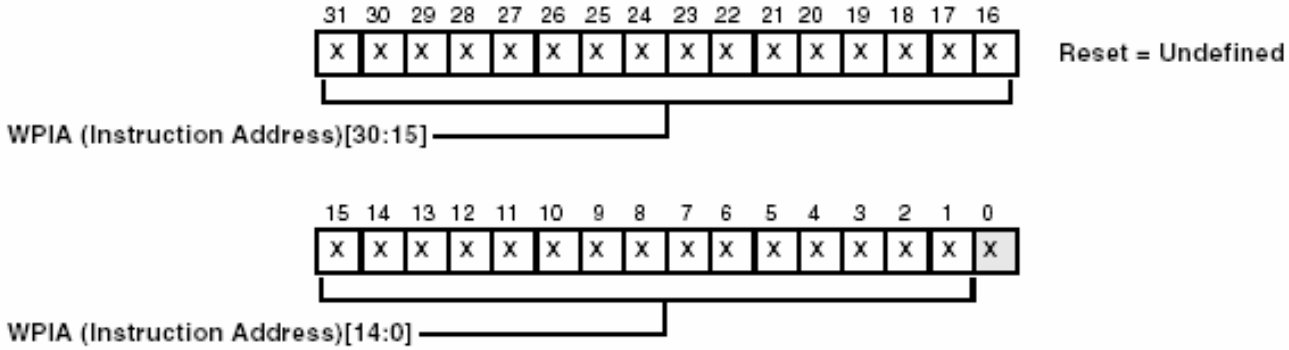
Watchpoint Control Registers

The Watchpoint Unit contains these MMRs, which are accessible in Supervisor and Emulator modes:

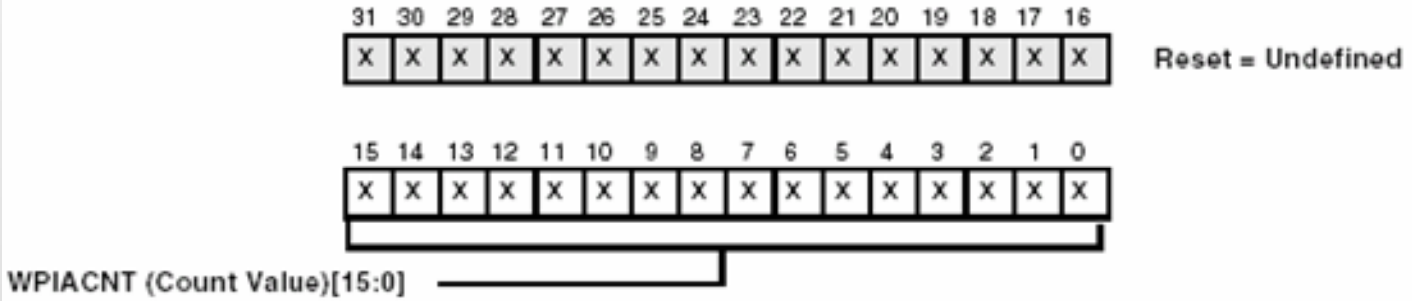
- The Watchpoint Status register (WPSTAT)
- Six Watchpoint Instruction Address registers (WPIA[5:0])
- Six Watchpoint Instruction Address Count registers (WPIACNT[5:0])
- The Watchpoint Instruction Address Control register (WPIACTL)
- Two Watchpoint Data Address registers (WPDA[1:0])
- Two Watchpoint Data Address Count registers (WPDACNT[1:0])
- The Watchpoint Data Address Control register (WPDACTL)

Watchpoint Instruction Address and Count Registers

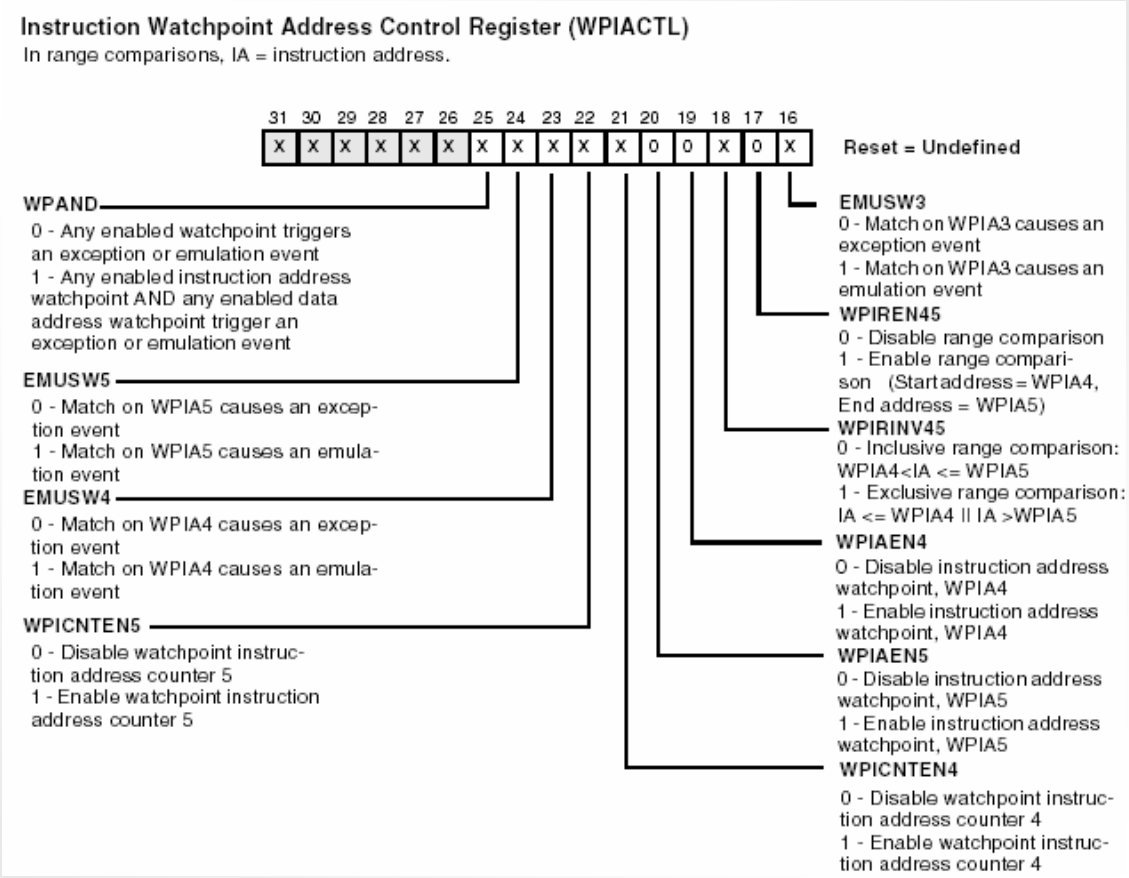
Instruction Watchpoint Address Registers (WPIAn)



Instruction Watchpoint Address Count Registers (WPIACNTn)



Watchpoint Instruction Address Control Register



- **Bits 0-15 of WPIACTL (not shown) control remaining instruction watchpoints**

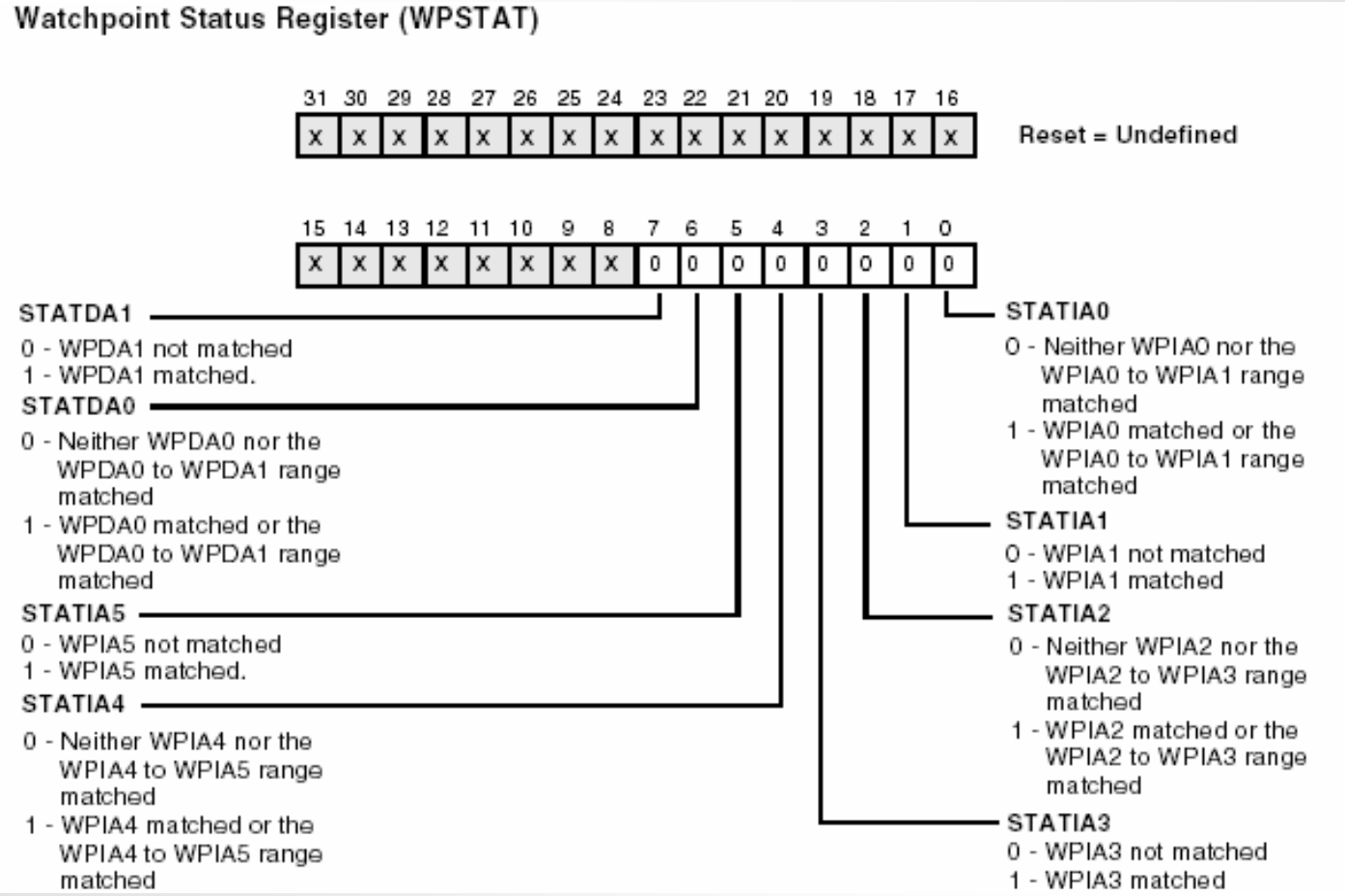
Watchpoint Data Registers

- **Analogous set of registers exist for two Data Watchpoints**

Each data watchpoint is controlled by four bits in the WPDACTL register:

Bit Names	Description
WPDMCCx	Determines whether the match should be on a read or write access.
WPDSRCx	Determines which DAG the unit should monitor.
WPDCNTENx	Enables the counter that counts the number of address matches. If the counter is disabled, then every match causes an event.
WPDAENx	Enables the data watchpoint activity.

Watchpoint Status Register



Status bits are sticky and are all cleared upon write of any value to the register

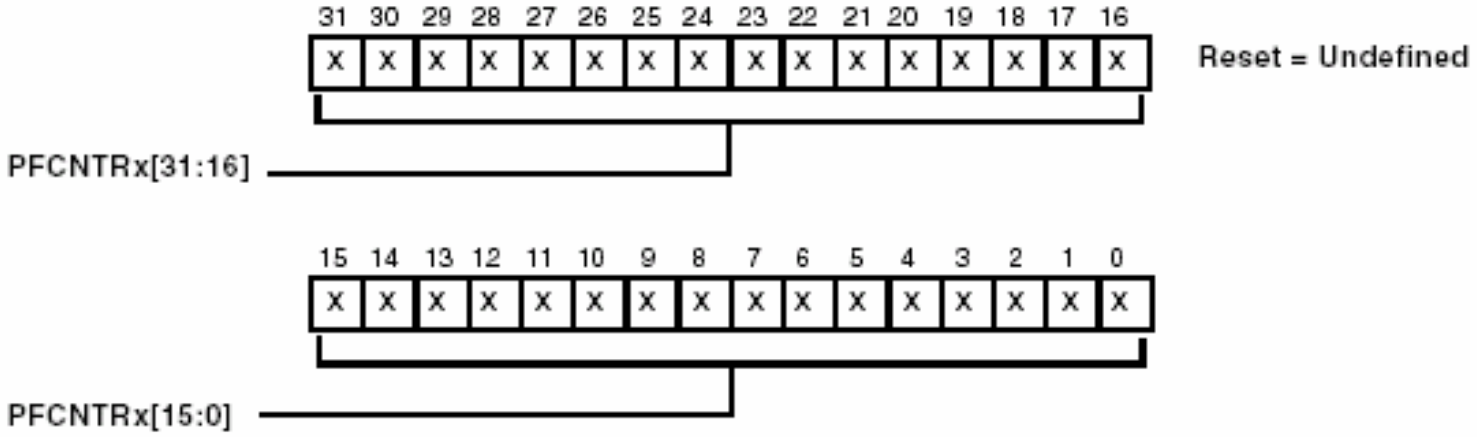
Performance Monitoring

- Dedicated registers available for system tuning and load balancing
- Includes two 32-bit counters, and registers to count number of cycles or occurrences of an event
  - Unit accesses (MAC0, MAC1, DAG0, DAG1)
  - Branches and exceptions
  - Memory conflicts
  - Loads/stores (8/16/32-bit)
  - Cache hits and misses
  - Interrupt latencies

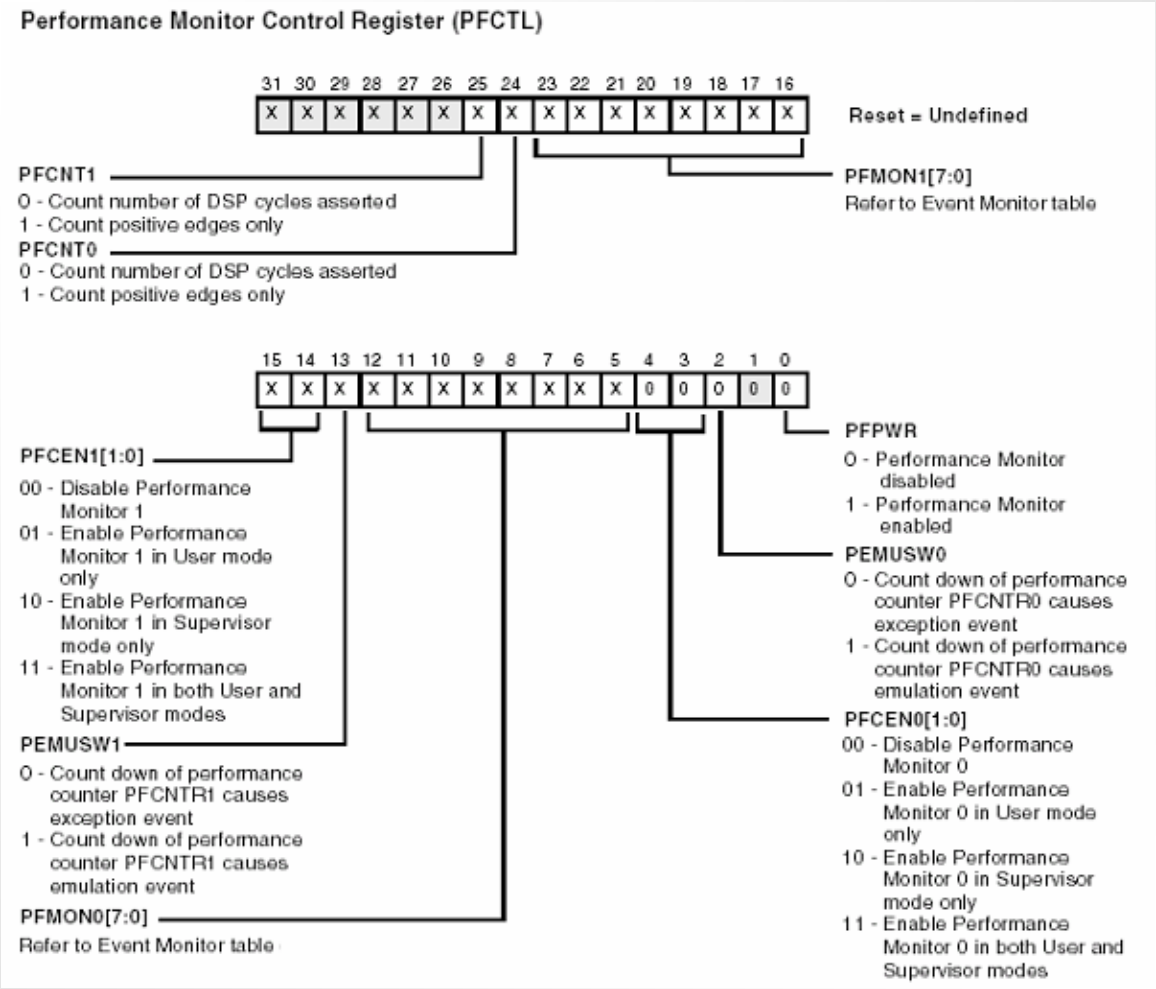
Performance Monitoring Registers

- Two Performance Monitor Counters exist

Performance Monitor Counter Registers (PFCNTRn)



Performance Monitor Control



What can be counted?

PFMONx Fields	Events That Cause the Count Value to Increment
0x00	Loop 0 iterations
0x01	Loop 1 iterations
0x02	Loop buffer 0 not optimized
0x03	Loop buffer 1 not optimized
0x04	PC invariant branches
0x06	Conditional branches
0x09	Total branches (calls, returns, branches, but not interrupts)
0x0A	Stalls due to CSYNC, SSYNC
0x0B	EXCPT instructions
0x0C	CSYNC, SSYNC instructions
0x0D	Committed instructions
0x0E	Interrupts taken
0x0F	Misaligned address violation exceptions
0x10	Stall cycles due to read after write hazards on DAG registers
0x13	Stall cycles due to RAW data hazards in computes
0x80	Processor stalls to memory
0x81	Data memory stalls to processor hidden by processor stall
0x82	Data memory store buffer full stalls

PFMONx Fields	Events That Cause the Count Value to Increment
0x83	Data memory write buffer full stalls due to high to low priority code transition
0x84	Data memory store buffer forward stalls due to lack of committed data from processor
0x85	Data memory fill buffer stalls
0x86	Data memory array or TAG collision stalls (DAG to DAG, or DMA to DAG)
0x87	Data memory array collision stalls (DAG to DAG or DMA to DAG)
0x88	Data memory stalls
0x89	Data memory stalls sent to processor
0x8A	Data memory cache fills completed to Bank A
0x8B	Data memory cache fills completed to Bank B
0x8C	Data memory cache victims delivered from Bank A
0x8D	Data memory cache victims delivered from Bank B
0x8E	Data memory cache high priority fills requested
0x8F	Data memory cache low priority fills requested
0x90	Code memory fetches postponed due to DMA collisions (minimum count of two per event)
0x91	Code memory TAG stalls (cache misses, or FlushI operations, count of 3 per FlushI). Note code memory stall results in a processor stall only if instruction assembly unit FIFO empties.
0x92	Code memory fill stalls (cacheable or non-cacheable). Note code memory stall results in a processor stall only if instruction assembly unit FIFO empties.
0x93	Code memory 64 bit words delivered to processor instruction assembly unit

Performance Monitoring: Cycle Counters

- All cycles are counted
- Counters stop during Emulation
- Can be read in all operating modes
- Can only be written in the Emulator or Supervisor modes
- Must be enabled to use

Example Code for Turning on Cycle Counters

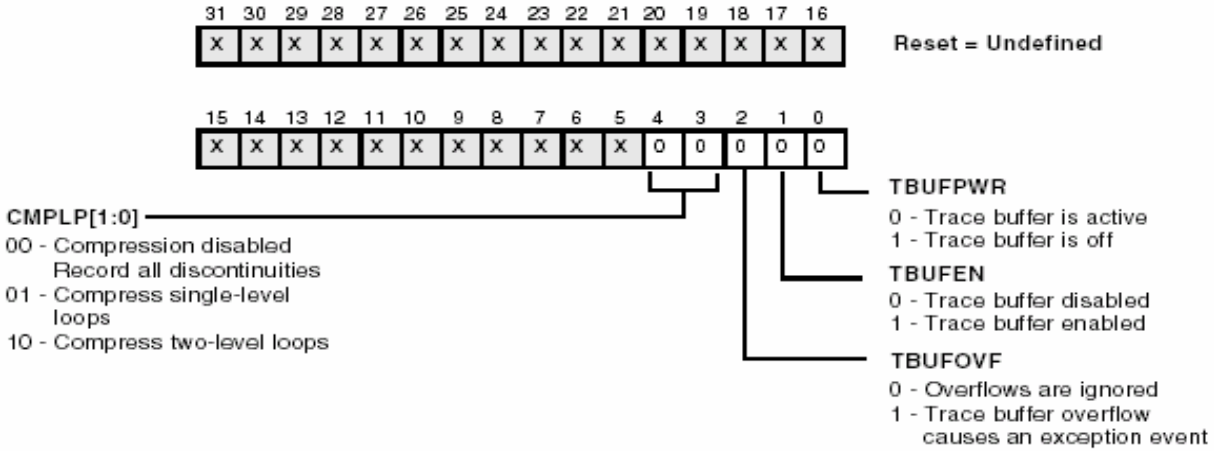
```
R2 = 0;
CYCLES = R2;
CYCLES2 = R2;
R2 = SYSCFG;
BITSET(R2,1);
SYSCFG = R2;
/* Insert code to be benchmarked here. */
```

Execution Trace

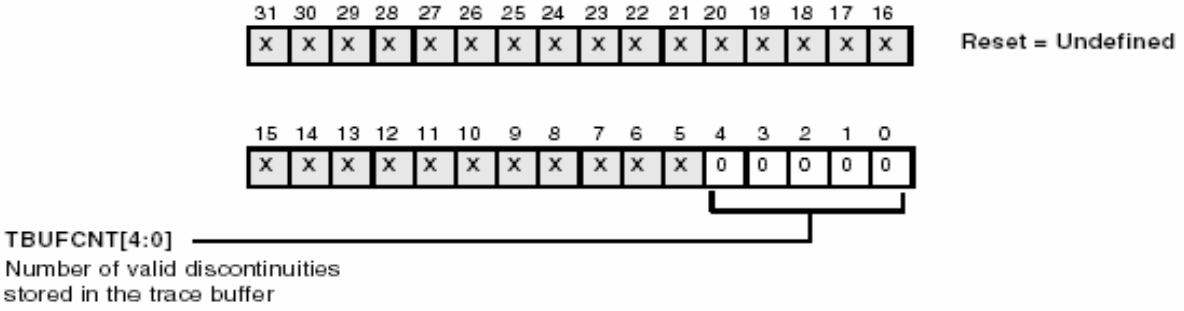
- 16-entry trace buffer stores changes in program flow (non-contiguous addresses)
  - Jumps
  - Calls
  - Interrupts
- Used for recent-history examination
- Zero-overhead hardware loops register only once in the trace buffer

Trace Buffer Registers

Trace Buffer Control Register (TBUFCTL)

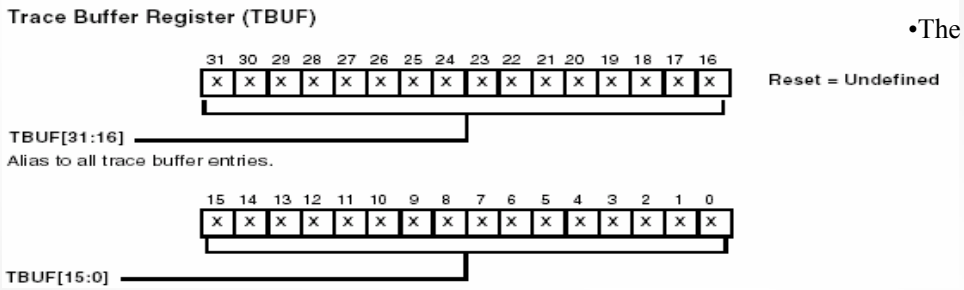


Trace Buffer Status Register (TBUFSTAT)



Accessing the Trace Buffer

- The first read returns the latest branch target address.
- The second read returns the latest branch source address.



Example Code for Recreating Execution Trace in Memory

```
[--sp] = (r7:7, p5:2); /* save registers used in this routine */
p5 = 32;  /* 32 reads are needed to empty TBUF */
   p2.l = buf;  /* pointer to the header (first location) of the
   software trace buffer */
   p2.h = buf;  /* the header stores the first available empty buf
   location for subsequent trace dumps */

p4 = [p2++];  /* get the first available empty buf location from
the buf header */
p3.l = TBUF & 0xffff;  /* low 16-bits of TBUF */
p3.h = TBUF >> 16;  /* high 16-bits of TBUF */

lsetup(loop1_start, loop1_end) lc0 = p5;
loop1_start: r7 = [p3];  /* read from TBUF */
loop1_end: [p4++] = r7;  /* write to memory and increment  */
[p2] = p4;  /* pointer to the next available buf location is
saved in the header of buf */
(r7:7, p5:3) = [sp++];  /* restore saved registers */
```